A Hybrid Approach for Spatio-Temporal Validation of Declarative Multimedia Documents

Joel A. F. dos Santos, Débora Christina Muchaluat-Saade, Cécile Roisin, Nabil Layaïda

A Hybrid Approach for Spatio-Temporal Validation of Declarative Multimedia Documents

JOEL A. F. DOS SANTOS, School of Computing and Informatics - CEFET/RJ
DÉBORA C. MUCHALUAT-SAADE, MídiaCom Lab, Univ. Federal Fluminense
CÉCILE ROISIN, Univ. Grenoble Alpes, Inria, LIG
NABIL LAYAÏDA, Inria, LIG, Univ. Grenoble Alpes

Declarative multimedia documents represent the description of multimedia applications in terms of media items and relationships among them. Relationships specify how media items are dynamically arranged in time and space during runtime. Although a declarative approach usually facilitates the authoring task, authors can still make mistakes due to incorrect use of language constructs or inconsistent or missing relationships in a document. In order to properly support multimedia application authoring, it is important to provide tools with validation capabilities. Document validation can indicate possible inconsistencies in a given document to an author so that it can be revised before deployment. Although very useful, multimedia validation tools are not often provided by authoring tools.

This work proposes a multimedia validation approach that relies on a formal model called *Simple Hypermedia Model* ($\mathcal{SHM}$). $\mathcal{SHM}$ is used for representing a document for the purpose of validation. An $\mathcal{SHM}$ document is validated using a hybrid approach based on two complementary techniques. The first one captures the document's spatio-temporal layout in terms of its state throughout its execution by means of a rewrite theory, and validation is performed through *model-checking*. The second one captures the document's layout in terms of intervals and event occurrences by means of *Satisfiability Modulo Theories* (SMT) formulas, and validation is performed through SMT solving. Due to different characteristics of both approaches, each validation technique complements the other in terms of expressiveness of $\mathcal{SHM}$ and tests to be checked.

We briefly present validation tools that use our approach. They were evaluated with real NCL documents and by usability tests.

CCS Concepts: • **Theory of computation → Program semantics**; **Program verification**;

Additional Key Words and Phrases: Interactive multimedia applications, multimedia authoring, multimedia document validation, spatio-temporal validation

We thank the support of CNPq and Inria in the context of Science without Borders Program. We also thank FAPERJ, CAPES and CNPq for their support.

Authors' addresses: J. A. F. Dos Santos, School of Computing and Informatics - CEFET/RJ, Av. Maracanã, 229, Bloco E, 5 andar - Maracanã, Rio de Janeiro, 20271-110, Brazil; email: jsantos@eic.cefet-rj.br; D. C. Muchaluat-Saade, MídiaCom Lab, Univ. Federal Fluminense, Rua Passo da Pátria, 156 - São Domingos, Niterói, 24210-240, Brazil; email: debora@midiacom.uff.br; C. Roisin and N. Layaïda, Univ. Grenoble Alpes, Inria, LIG, Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000 Grenoble, France; emails: {cecile.roisin, nabil.layaida}@inria.fr.

1551-6857/2018/10-ART86 $15.00
https://doi.org/10.1145/3267127

ACM Reference format:
Joel A. F. Dos Santos, Débora C. Muchaluat-Saade, Cécile Roisin, and Nabil Layaïda. 2018. A Hybrid Approach for Spatio-Temporal Validation of Declarative Multimedia Documents. *ACM Trans. Multimedia Comput. Commun. Appl.* 14, 4, Article 86 (October 2018), 24 pages.
https://doi.org/10.1145/3267127

1 INTRODUCTION

Multimedia applications are present in our daily life as web pages, digital TV programs, smartphone applications, advertising screens, and so on. They consist of media items synchronized in time and space, allowing viewer interactions in one or more display devices. Multimedia applications are usually defined with a domain-specific language, called a *multimedia authoring language*. Such languages focus on the definition of media items as part of a multimedia application and their presentation order in time and space [7, 24]. A subset of those languages uses a declarative approach, usually based on XML, for providing high-level constructs for defining multimedia applications. They simplify the definition of a multimedia application since they emphasize its description rather than implementation details. Examples of declarative multimedia authoring languages are *HyperText Markup Language* (HTML5) [40], *Synchronized Multimedia Integration Language* (SMIL) [38] and *Nested Context Language* (NCL) [26].

A declarative multimedia document is the one specified using a declarative authoring language. It is composed of a set of media items and the relationships among them, which define their presentation order in time and space. In this article, we refer to a set of document relationships as a *layout*. A document's *temporal layout* indicates a media item's presentation order in time and also in reaction to viewer interaction, while its *spatial layout* indicates a media item's presentation position on a screen or a set of screens. A combination of both is called the document *spatio-temporal layout*. This article focuses on declarative multimedia documents and declarative multimedia authoring languages. For simplicity, we shall refer to these simply as *document* and *authoring language*, respectively.

Besides providing a clear separation between the application description and its implementation, a positive side effect of declarative authoring languages is to diminish the authoring effort, particularly for authors with little or no programming skills. This is desirable since documents can be used in different areas, such as web [38], digital TV [1], and IPTV [26], and by different author profiles, such as program developers and content producers.

On the other hand, constructs of an authoring language have a complex semantics, given that they abstract several definitions that the author would have to provide if he or she had used a programming language. Large-scale documents, with many components organized within quite rich and complex structures, may fall victim to poor visibility, hidden dependencies, and error-proneness [23]. The resulting spatio-temporal layout perceived by an author may not fit his mental representation due to the incorrect use of language constructs [29]. We call such mismatch between the perceived spatio-temporal layout and the author's mental representation an *undesired behavior*. As discussed in Kostalas et al. [29], it is important that authors are able to identify undesired behaviors in a document prior to its deployment.

Usually, authors test their documents by executing them in an attempt to identify undesired behaviors. This approach is not *complete* since not all possible behaviors are explored, especially when viewer interaction is possible, thus leading to a huge amount of presentation results. This approach is not *effective* since several executions would be necessary for the verification of undesired behaviors. Moreover, multimedia documents can be automatically generated inside a production pipeline. In this case, simulating a document execution before deployment would be costly, if possible.

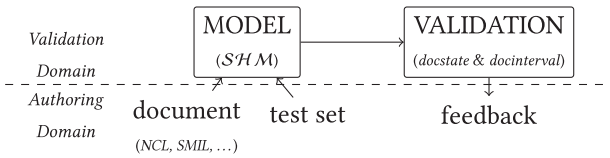


Fig. 1. Validation approach overview.

In this work, we present a hybrid approach for the automatic validation of multimedia documents. By validation, we mean the process of checking a set of tests over a document. Each test represents an *expected behavior* an author wants to check or an *unexpected behavior* to be avoided.

Figure 1 depicts an overview of the validation approach we propose. Our approach relies on a formal model for representing the spatio-temporal layout of multimedia documents named *Simple Hypermedia Model* ($\mathcal{SHM}$). A declarative document is translated into $\mathcal{SHM}$ to perform its validation. Validation is done by the application of a hybrid approach based on two techniques—document-state-based validation (*docstate*) and interval-based validation (*docinterval*)—considering a test set. Validation test results are presented as feedback to the author, informing success or errors and allowing, as stated in Kostalas et al. [29], the author to revisit the application being created to adjust the spatio-temporal specification to eliminate *undesired behaviors*.

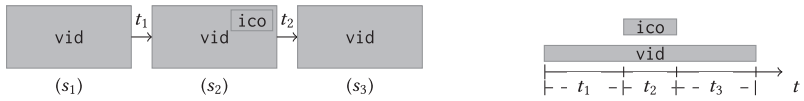
It is important to support different authoring languages. Therefore, the validation approach we present (model and techniques) considers different conceptual entities usually found in distinct authoring languages, such as events (either deterministic or indeterministic), compositions, constraint relations, and causal relations. In addition, given its support for indeterministic events, our validation approach is able to handle interactive documents, thus considering every different outcome generated by viewer interactions. It is worth mentioning, however, that some authoring languages provide the possibility of using auxiliary scripting languages, such as Lua [25] in NCL and JavaScript in HTML. The validation approach proposed in this work does not tackle such scripting languages, focusing only on the authoring language itself.

Moreover, according to the authoring language, a document may be validated at the same time it is created. For example, in a constraint-based authoring approach, such as in Madeus [27], where constraints are used to define a document's temporal layout, relations may be validated as they are created. On the other hand, in an event-based language, such as NCL [26], where causal relations over event occurrences are used for defining the document's temporal layout, validation may have to wait for the creation of a set of relations before being performed. The approach we propose follows the latter path, requiring the creation of a set of relations before validation is performed.

The contribution of this article is two-fold. We present a formal model for representing the spatio-temporal layout of multimedia documents and a hybrid validation approach combining two different validation techniques supporting the validation of different kinds of *expected behaviors*. Our validation approach is implemented by authoring tools that provide NCL document validation.

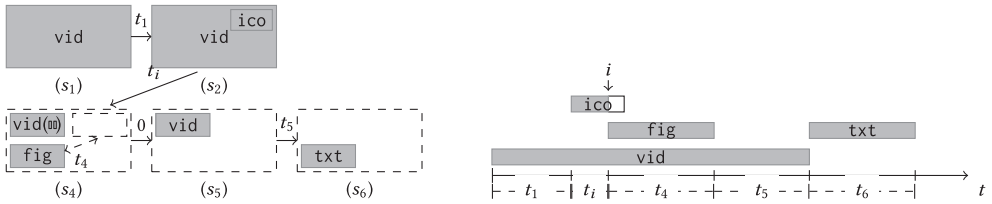
This article is a significantly revised and extended version of earlier works [14–17]: (i) it includes an updated definition of the model $\mathcal{SHM}$ that encompasses both event- and interval-based semantics, (ii) it proposes a hybrid validation approach built from this semantic, and (iii) it discusses the *docinterval* validation approach, which was briefly outlined in dos Santos et al. [17].

The remainder of this article is structured as follows. Section 2 discusses spatio-temporal validation. Section 3 presents related work considering the validation of multimedia documents. Section 4 describes model $\mathcal{SHM}$. Section 5 discusses how an author describes a set of tests to be performed over a document. Section 6 presents our hybrid validation approach. It also discusses how $\mathcal{SHM}$ is used during validation and presents validation tools based on our approach,



(a) spatial layout along time (b) temporal layout

Fig. 2. Spatial-temporal layout without interaction.



(a) spatial layout along time (b) temporal layout

Fig. 3. Spatial-temporal layout with interaction.

together with usability tests. Section 7 discusses the use of each technique, with its advantages and disadvantages. Section 8 concludes this article and indicates future work.

2 SPATIO-TEMPORAL VALIDATION

A document's spatio-temporal layout is commonly defined as two sets of definitions. In the temporal axis, media items are placed in time either with absolute values or in relation to other media items or event occurrences, such as viewer interaction. In the spatial axis, as presented in Hardman [24], media items are placed in relation to the screen, another media item, or into predefined channels.

Media item position and size may change over time, as an animation. Some authoring languages allow the author to define animations in response to event occurrences. Animations may move a media item around the screen by changing, for example, its *left/top* attributes, or scale a media by changing, for example, its *width/height* attributes. It is worth mentioning that animations may occur incrementally over a time interval.

As an example, consider the spatio-temporal layout of document d in Figures 2 and 3. Although simple, this example presents several characteristics found in real-world multimedia applications. The example is composed of four media items: the main video (`vid`), an advertisement icon (`ico`), an animated figure (`fig`), and a casting text (`txt`).

Figures 2(a) and 3(a) present d's spatial layout along time. In both figures, a rectangle with a dashed border represents a screen region (at some moment), and rectangles filled with solid colors represent a visible media item. Arrows between two screens represent a time lapse, and dashed arrows (see Figure 3(a), screen s_4) represent an animation, which can be done instantaneously or incrementally over a time period. In the latter, animation duration is presented over the arrow.

Two different outcomes are possible for this example, depending on viewer interaction with `ico`. In the case where the viewer does not interact, it behaves as depicted in Figure 2(a) by the sequence of screens s_1, s_2, and s_3. `vid` is presented on the whole screen (s_1) when the execution begins. After t_1 time units, `ico` is presented on the upper-right corner (s_2) for t_2 time units. After that, `vid` keeps being presented for more t_3 time units until it ends (s_3). The temporal layout for the first outcome is depicted in Figure 2(b).

Considering the second outcome, the example behaves as depicted in Figure 3(a) by the sequence of screens s_1, s_2, s_4, s_5, and s_6. When `ico` is presented (s_2) and the viewer interacts with it after an unknown time t_i ($t_i < t_2$), `vid` is downsized and repositioned on the upper-left corner. This change in size and position is instantaneous. Moreover, *vid* presentation is paused (▯▯) and `fig` is presented on the lower-left corner (s_4). As soon as it starts its presentation, `fig` moves from its initial position to the upper-right corner and back in t_4 time units. When it comes back to its original position, its presentation ends, and `vid` presentation is resumed (s_5). After t_5 time units, `vid` presentation ends and `txt` is presented on the lower left corner for t_6 time units (s_6). The temporal layout for the second outcome is depicted in Figure 3(b), where viewer interaction is represented as ↓. This example will be used for illustrating different types of tests to be checked.

Usually, the validation of a document's spatial layout is performed separately from the temporal layout [6, 28, 30]. Spatial validation is usually offered by indicating possible overlapping of media items. This is performed by a two-step approach, where first it verifies if media items overlap based on their initial position and then it verifies if they are presented together.

Considering document d, tests with media items `ico` and `txt` can be described with temporal tests or spatial tests separately, as they do not change their position over time. For example, the author may want to ensure that `ico` is presented before `txt` or that `ico` and `txt` are not presented simultaneously. Moreover, the author may wish to ensure that `ico` is positioned above `txt` or that they should have different sizes. Since the spatial layout of both media items does not change over time, spatial validation can be done considering their initial position.

On the other hand, tests with media items `vid` and `fig`, corresponding to screen s_4 in Figure 3(a), are not so simple as statically validating the spatial layout and, in parallel, validating the temporal layout. For example, the author may want to ensure that, at some point while moving across the screen, `fig` will overlap `vid`. This requires verifying if, in at least one moment during the document execution, `fig` and `vid` will overlap. Such a test has to be encoded by composing temporal and spatial tests. It is also possible for the author to create tests relating to sequential changes, such as "*is* `fig` *above* `vid`*, and then at the right side of* `vid`*?*" To specify such a test, one should use a temporal connector between the two tests (`fig` is above `vid` and `fig` at the right side of `vid`).

These examples illustrate tests where (i) the spatial and temporal layouts are independent or (ii) the spatial layout is dependent on the temporal one.

A purely temporal validation acts just in the temporal axis, verifying the consistency of temporal relationships among media items. On the other hand, a purely spatial validation acts just in the spatial axis, verifying the consistency of spatial relationships among media items. Validation in both temporal and spatial axes can be done in two ways. In the first, validation of the spatial axis is performed separately from the temporal axis. It is usually offered by indicating possible overlapping of media items according to their initial positions and sizes and their organization in time. In the second, validation of both axes is performed together, considering changes in space over time, which we call spatio-temporal validation. However, when media item positions may change over time, as it can be seen in Figure 3(a), reasoning about space separately from time gives incorrect results. It is important, therefore, to consider the spatio-temporal layout of a document while validating it, which is supported by our proposal.

In addition to the kind of validation provided (temporal, spatial, and spatio-temporal), validation also may be carried out by two different approaches: (i) based on the document state during its execution or (ii) based on constraints. The first one models the document state and how it changes throughout document execution. This is done by representing a document as a state machine or by providing the semantic description of constructs of a given authoring language in a given logic. The document being created is either transformed to such representation or created directly in it. Validation is then performed by reachability analysis over the state machine or by axiom

Table 1. Related Work Classification

	[21]	[22]	[9]	[28]	[6]	[30]	[20]	[32]	$\mathcal{SHM}$
Temporal	✓	✓	✓						
Temporal + Static spatial				✓	✓	✓			
Spatio-temporal							✓	✓	✓
State reachability	✓	✓	✓	✓					✓
Constraint checking					✓	✓	✓	✓	✓

application in the given logic, respectively. Thus, it is possible to verify if a document may reach a given state where a media item is being presented, for example.

The second approach models document relations as constraints. This is done by providing either a transformation for documents in a given authoring language into a set of constraints or using an authoring language based on constraints. Validation in both cases is provided by checking the consistency of the constraint set as a whole. Thus, it is possible to verify if constraints hold together.

3 RELATED WORK

The literature is rich in discussions about document validation. We classify the related work presented in this section according to the reasoning principle applied for document validation. The first group of papers [9, 21, 22, 28] handles validation based on the document state. The second group of papers [6, 20, 30, 32] handles validation based on constraints. We also classify them according to their coverage in either the spatial axis or temporal axis. Table 1 presents related work classification.

Felix [21] presents an approach for the validation of temporal properties of NCL documents using model checking techniques. NCL documents are translated into a state machine–like representation that indicates the document's temporal layout. The transformation creates a state machine for each media item and a synchronizer machine for each relationship. The latter is used for tying together the occurrence of events in media state machines. The validation of an NCL document is done over that representation. The author describes tests as temporal logic formulas.

Gaggi and Bossi [22] define a formal semantics for SMIL through a set of inference rules inspired by Hoare logic. The rules describe the document state before and after the execution of a given SMIL construct. Thus, in the authoring phase, the structure of an SMIL document may be enriched with assertions expressing temporal properties. Document validation is performed by the application of axioms, also defined in the proposed semantics, that verify if a given construct or set of constructs correctly changes the document state. Otherwise, it presents to the author the problem found so it can be corrected.

Bouyakoub and Belkhir [9] present an incremental authoring tool for SMIL documents called SMIL Builder. Whenever an author changes a document, SMIL Builder checks if this change will cause an inconsistent temporal layout. If so, the change is rejected and an error report is presented to the author. Both incremental editing and consistency checking are supported by the H-SMIL-Net model [8]. H-SMIL-Net extends Petri Nets in order to fully represent SMIL temporal characteristics. Inconsistencies are detected either by construction (e.g., when a transition has more arcs than possible) or by verification of transition firing times in the model.

As can be seen in Table 1, some works [9, 21, 22] present a purely temporal approach, where the validation of a document is performed by investigating its state over its execution. In Felix [21], it is done by reachability analysis, in Bouyakoub and Belkhir [9] it is done by verifying the

consistence of the underlying Petri Net, and in Gaggi and Bossi [22] by analyzing if the document state changes according to some axioms. Document spatial layout validation is not discussed in those papers.

Júnior et al. [28] use a model-driven approach for the validation of NCL documents. The validation is achieved by transforming an NCL document into a Kripke structure. The validation uses a model-checking tool and temporal logic formulas representing the properties to be validated. Spatial validation is briefly discussed in Picinin [28] and is performed over the document's initial positioning.

Bertino et al. [6] propose an authoring model based on constraints. A document in that model consists of several topics, where each topic is composed of semantically related media items. The system automatically groups media items into topics according to constraints defined by the author. The system enlarges the set of constraints with others that, even when not explicitly defined, are consequences of the constraints defined by the author. Consistency checking is then performed over the constraint set. If an inconsistency arises, the system applies relaxation techniques to reduce the constraint set to a consistent one. When this is not possible, an author review is required.

Laborie et al. [30] present an approach for the automatic adaptation of a document layout according to an exhibition device. It creates an abstract description of a document as a set of objects and constraints representing temporal and spatial relations among objects. It also considers a profile comprising device constraints and viewer preferences. According to the set of potential document executions $\mathcal{M}_s$, given by the abstract description, and the set of potential executions $\mathcal{M}_p$, given by the profile, the adaptation process calculates $\mathcal{M}_s \cap \mathcal{M}_p$ to determine if some adaptation is required. In case the document has to be adapted, the goal is to change a document's relations such that it now complies with the profile, and the (behavioral) distance from the previous declaration is minimum.

As can be seen in Table 1, some works presented [6, 28, 30] cover both temporal and spatial dimensions. In the first one, document validation is performed by model-checking, and, in the second and third, it is performed by consistency checking over a set of constraints. However, in those works, the spatial dimension is static since spatial constraints do not change over time. Moreover, reasoning about time and space is performed as two separate problems.

Elias et al. [20] and Ma and Shin [32] also propose an authoring model based on constraints. Both works provide constraints for modeling the temporal and spatial relations. Constraints have an associated priority value. Spatial constraints are associated with an interval, such that a given (spatial) constraint has to be satisfied inside that interval. Both approaches transform the constraint set to a graph. Completeness checking is done by checking if the graph is connected. Consistency checking is done by finding the minimum spanning tree T in the constraint graph. Constraints that create inconsistencies (i.e., when not contained in T) are removed according to their priority values or by author choice. Elias et al. [20] provide relaxation techniques to be applied when two constraints have the same priority, while Ma and Shin [32] provide an approach to *relax* constraints considering others not contained in T. One should note that, although these works [20, 32] do not define spatial attributes in function of time, they represent examples of constraints parameterized by time.

As can be seen in Table 1, together with our proposed approach, Elias et al. and Ma and Shin [20, 32] are the only ones to provide a truly spatio-temporal validation, given that spatial constraints may change over time. In previous works [15–17], we presented a document validation approach based on the first version of the $\mathcal{SHM}$ model. It uses the document-state-based validation technique presented in Section 6.1 and provides validation in both time and space as presented in dos Santos et al. [17]. This work extends our previous validation approach by providing a hybrid validation approach based on both techniques, document state and constraints. To the best of our

knowledge, this is the first attempt at combining these two approaches thanks to our $\mathcal{SHM}$ model that includes both semantics.

4 VALIDATION MODEL

The validation approach proposed in this paper is based on a formal model called *Simple Hypermedia Model* ($\mathcal{SHM}$). It captures the essence of multimedia document spatio-temporal behavior, hiding the specific syntax and details of an authoring language. We call the representation of a document d in model $\mathcal{SHM}$ an $\mathcal{SHM}$ spatio-temporal layout.

Definition 4.1 ($\mathcal{SHM}$ spatio-temporal layout). An $\mathcal{SHM}$ spatio-temporal layout is defined as a tuple

$$\ell = \langle \Omega, \Gamma \rangle, \tag{1}$$

where Ω is a set of attributes and Γ is a set of binary relations between attributes.

$\mathcal{SHM}$ represents a document execution by a set of attributes. They characterize the execution of document parts, called here *fragments*, holding information about them. Attribute values change throughout document execution according to the semantics of the fragment they characterize. Document fragments may be (i) document compositions as a whole, (ii) document variables, or (iii) media items and their subparts (anchors) in time or space.

Definition 4.2 ($\mathcal{SHM}$ attribute). An $\mathcal{SHM}$ attribute is defined as a tuple

$$\omega = \langle id, val \rangle, \tag{2}$$

where id is an attribute identifier and $val \in Bool \cup \mathbb{N} \cup String \cup State$, with sets $Bool = \{\top, \bot\}$ and $State = \{occurring, paused, sleeping\}$; that is, the attribute value can be either a Boolean, a number, a string, or a state.

A set of attributes is used to represent a given facet of a document fragment execution: its presentation, attribution, or viewer interaction. An attribute whose value is of sort *State* has embedded semantics, such that changes in its value are signaled as event occurrences (denoted ε). Thus $\mathcal{SHM}$ relations may be described over event occurrences or attribute values.

Definition 4.3 ($\mathcal{SHM}$ relation). Relations in $\mathcal{SHM}$ may be a causal relation or a constraint relation. A causal relation has the general form

$$\langle p,\ \varepsilon(\omega_k),\ \varepsilon(\omega_l),\ \varphi(\omega_m, \ldots, \omega_n) \rangle, \tag{3}$$

and a constraint relation has the general form

$$\langle p,\ \delta(\omega_k, \ldots, \omega_l),\ \varphi(\omega_m, \ldots, \omega_n) \rangle, \tag{4}$$

where $p \in \mathbb{N}$ represents a priority value for the relation, ε is an event occurrence over state attributes ω_k and ω_l, δ is a statement, and φ is a precondition over attributes $\omega_i \in \Omega$. In both relation types, p and φ may be omitted and are assumed to be the biggest priority value and $\top$, respectively.

Causal relations specify that whenever a condition event ($\varepsilon(\omega_k)$) occurs, another action event ($\varepsilon(\omega_l)$) will occur. If the condition event does not occur during document execution, the action event will not occur either. Constraint relations describe a statement ($\delta(\omega_k, \ldots, \omega_l)$) that must hold during document execution. Relations may specify a precondition over attribute values ($\varphi(\omega_m, \ldots, \omega_n)$), such that a relation will only be considered if its precondition is evaluated to true. Moreover, relations are applied according to their priority value.

The semantics of document fragments are translated into $\mathcal{SHM}$ relations that change attribute values according to the semantics defined by the authoring language. Moreover, $\mathcal{SHM}$ relations

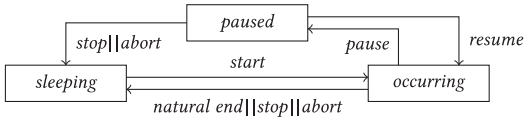


Fig. 4. NCM State machine.

will also capture the behavior of relationships defined in the multimedia document. Both the temporal and spatial layouts of a document are described in $\mathcal{SHM}$ by the specification presented in Definitions 4.1–4.3. The following sections discuss how $\mathcal{SHM}$ represents fragments in both time and space.

4.1 Temporal Axis

To characterize fragments in the temporal axis, $\mathcal{SHM}$ attributes are used to represent a fragment *duration*, *occurrences* number, *state,* and its *existence* in the temporal layout (i.e., if it is ever executed). Attribute values representing the fragment duration and occurrences number are of set $\mathbb{N}$, the one representing a fragment state is of set *State* and the existence attribute value is of set *Bool*.

$\mathcal{SHM}$ considers the set of states *State* = {*occurring*, *sleeping*, *paused*}. From now on, we call such an attribute a *state-attribute*. The state-attribute value changes according to the state machine presented in Figure 4.

The state machine presented in Figure 4 is defined by the Nested Context Model (NCM) [36], upon which NCL is based. NCM events have a duration and a type. A presentation event represents a document fragment presentation, an attribution event represents a document variable change, and a selection event represents viewer interaction with a media item or a subpart of it.

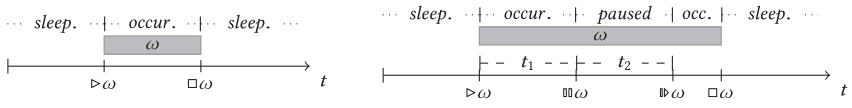
An NCM event initiates at the *sleeping* state. As the document presentation unfolds, it will eventually change to the *occurring* state. It remains in the *occurring* state for a given period of time, given by its duration. After its duration, it goes back to the *sleeping* state. This behavior is called its *natural end*. It can also go back to the *sleeping* state before its *natural end* as a result of a forced end (*stop* or *abort*). Presentation event durations are defined according to media item durations. Attribution and selection event durations are equal to zero.

State-attributes, representing either a fragment presentation, attribution, or selection, will behave the same way as NCM events. An $\mathcal{SHM}$ event occurrence, therefore, represents a transition in the NCM state machine; that is, a change in the value of a state-attribute. They are represented by the following symbols: ▷ (*start*), □ (*natural end* or *stop*), ⊠ (*abort*), ▯▯ (*pause*), and ⧐ (*resume*).

Relations in time may be defined based on event occurrences as either a causal relation or as a constraint relation. In causal relations, event occurrences are represented by the juxtaposition of an event symbol and state-attribute id. It can either represent a state-attribute change that has already occurred (when it appears as the relation condition) or a state-attribute change to be triggered (when it appears as the relation action). For example, the following causal relation

$$\langle \square vid_{(p,s)},\ \triangleright txt_{(p,s)},\ ico_{(s,n)} > 0 \rangle \tag{5}$$

represents a desired behavior of the running document shown in Figures 2 and 3, where fragment `txt` starts its presentation when fragment `vid` ends, given that the viewer has interacted with `ico`. In Equation (5), $vid_{(p,s)}$ and $txt_{(p,s)}$ are the id of the state-attributes representing the presentation state of fragments `vid` and `txt`, respectively. Thus, $\square vid_{(p,s)}$ represents the natural end of `vid`'s presentation and $\triangleright txt_{(p,s)}$ the start of `txt`'s presentation. Attribute $ico_{(s,n)}$ represents the number of times the viewer selected fragment `ico`. As one notes from the previous example, we use the following naming scheme for $\mathcal{SHM}$ attributes $f_{(x,y)}$, where f is a fragment identifier; x indicates the fragment facet being represented, which is either p for *presentation*, a for *attribution,* or s for



(a) Interval example

(b) Interval example with pause

Fig. 5. Interval examples.

selection; and y indicates the kind of value the attribute holds, which is either d for its *duration*, n for its *occurrence* number, s for its *state,* or e for its *existence* in the temporal layout.

In constraint relations, the moment in time when a state change occurs is represented by the juxtaposition of an event symbol and a state-attribute id. For example, the following constraint relation

$$\langle \triangleright ico_{(p,s)} \leq \triangleright ico_{(s,s)} \wedge \ \square ico_{(s,s)} \leq \square ico_{(p,s)},\ ico_{(s,e)} == true \rangle, \tag{6}$$

based on the document shown in Figures 2 and 3, specifies that fragment `ico` can be selected only while it is being presented, in the case where it occurs. In Equation (6), $ico_{(p,s)}$ and $ico_{(s,s)}$ are the id of the state-attributes representing, respectively, the presentation and selection state of fragment `ico`. Thus, $\triangleright ico_{(p,s)}$ represents the start of `ico`'s presentation and $\triangleright ico_{(s,s)}$ the start of `ico`'s selection.

The use of event occurrences in constraint relations, as seen in Equation (6), is possible due to the facility of projecting a state-attribute value in the time axis, such that an interval indicates when it has a certain value along the document presentation. Figure 5 presents the projection of a state-attribute ω in the time axis.

In Figure 5(a), interval endpoints ($\triangleright\omega$ and $\square\omega$) represent begin and end times of a given fragment (represented by ω) according to its definition in a document. As can be seen in the figure, attribute ω's value is equal to *occurring* inside the interval between $\triangleright\omega$ and $\square\omega$ and is equal to *sleeping* elsewhere. Figure 5(b) modifies the example shown in Figure 5(a) such that a given fragment is paused for t_2 seconds, t_1 seconds after its begin time, according to its definition in a document. In such a case, attribute ω's value is equal to *paused* inside the interval between $\blacktriangleright\!\!\blacktriangleright\omega$ and $\blacktriangleright\!\!\blacktriangleright\omega$.

The $\mathcal{SHM}$ temporal layout for the document shown in Figures 2 and 3 is as follows:

$$\ell = \langle \{vid_{(p,s)}, ico_{(p,s)}, fig_{(p,s)}, txt_{(p,s)}, ico_{(s,s)}, ico_{(s,e)}, ico_{(s,n)}\}, \{\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5, \gamma_6, \gamma_7\} \rangle \tag{7}$$

$$\gamma_1 = \langle \triangleright ico_{(p,s)} = \triangleright vid_{(p,s)} + t_1 \rangle \tag{8}$$

$$\gamma_2 = \langle \triangleright ico_{(p,s)} \leq \triangleright ico_{(s,s)} \wedge \ \square ico_{(s,s)} \leq \square ico_{(p,s)}, ico_{(s,e)} == true \rangle \tag{9}$$

$$\gamma_3 = \langle 1,\ \square ico_{(s,s)},\ \square ico_{(p,s)},\ ico_{(s,e)} == true \rangle \tag{10}$$

$$\gamma_4 = \langle 2,\ \square ico_{(s,s)},\ \blacktriangleright\!\!\blacktriangleright vid_{(p,s)},\ ico_{(s,e)} == true \rangle \tag{11}$$

$$\gamma_5 = \langle 3,\ \square ico_{(s,s)},\ \triangleright fig_{(p,s)},\ ico_{(s,e)} == true \rangle \tag{12}$$

$$\gamma_6 = \langle \square fig_{(p,s)},\ \blacktriangleright\!\!\blacktriangleright vid_{(p,s)} \rangle \tag{13}$$

$$\gamma_7 = \langle \square vid_{(p,s)},\ \triangleright txt_{(p,s)},\ ico_{(s,n)} > 0 \rangle \tag{14}$$

It is worth mentioning that, for simplicity, relations specifying a change in state-attribute values from *occurring* back to *sleeping* after the fragment duration are not presented in the previous example.

4.2 Variable Handling

Multimedia documents often use variables to store information useful for building their spatio-temporal layout [37], such as fragment position, visibility, and the like. For example, in the

document presented in Figures 2 and 3, a variable is used to store the number of times the viewer interacted with the icon. $\mathcal{SHM}$ represents variables by attributes that hold their values. During document execution, variable values may change as a reaction to event occurrences. Given that document variable changes may be used in document relations, $\mathcal{SHM}$ also associates state-attributes to them, as discussed in Section 4.1.

$\mathcal{SHM}$ captures the behavior of a change in the value of document variables by event $\circ$, which is followed by the value change to be performed. The following example presents a relation that increases the number of occurrences of fragment `ico` selection whenever its selection ends:

$$\langle \Box ico_{(s,s)},\ \circ ico_{(s,n)} = ico_{(s,n)} + 1 \rangle. \tag{15}$$

Suppose a document defines a variable `var` such that a change in its value triggers the application of causal relations specified in the document. This variable, therefore, shall be represented in $\mathcal{SHM}$ by two attributes, one representing its value and another its state. The following example modifies the one presented in Equation (15), such that variable `var` value is changed instead of $ico_{(s,n)}$.

$$\langle 1, \Box ico_{(s,s)},\ \circ var_{(a,v)} = ''foo'' \rangle \tag{16}$$

$$\langle 2, \Box ico_{(s,s)},\ \triangleright var_{(a,s)} \rangle. \tag{17}$$

The reader should note that, in addition to the relation for changing the value of variable `var` (attribute $var_{(a,v)}$), a second one changes its attribution state from *sleeping* to *occurring* (attribute $var_{(a,s)}$). It is worth mentioning that, given the fact that attribution duration is equal to zero, state-attribute $var_{(a,s)}$ value will change back to *sleeping* immediately.

4.3 Spatial Axis

In the spatial axis, document fragments are considered as rectangular regions. The spatial definition for fragments is specified over their projections in three spatial axis $\{x, y, z\}$, where z is used for overlapping order.

$\mathcal{SHM}$ attributes hold spatial information about document fragments, as discussed in Section 4.2. According to the naming scheme presented in Section 4.1, in the spatial axis, attributes are named as follows: $f_{(a,i)}$, where f is a fragment identifier, a is one of the spatial axis $\{x, y, z\}$, and i gives information about the fragment projection in that axis (b for the projection begin, c for the projection center, e for the projection end, and s for the projection size).

Fragment position and sizes in the spatial axis may be specified by absolute values or by constraint relations. In the first, the value for the attributes holding the fragment position and size is specified *a priori* and does not change during document execution. In the second, a constraint or causal relation sets their values.

Considering the document shown in Figures 2 and 3, the following equation presents an example of a constraint stating that `fig` and `txt` should have the same size:

$$\langle fig_{(x,s)} = txt_{(x,s)} \ \wedge\ fig_{(y,s)} = txt_{(y,s)} \rangle. \tag{18}$$

4.4 Spatio-Temporal Relations

Authoring languages usually provide the possibility of changing document fragment positions and sizes during execution. A common way to achieve this is to declare relations that change fragments position and size in response to an event occurrence. This change can be either *discrete*, or *incremental* over a time interval. In the latter, the relation provides, together with the new values, the duration of the change and the increment by which values have to be changed.

$\mathcal{SHM}$ captures this behavior in two ways: (i) as discussed in Section 4.3, attributes are used to store the position of fragments, and (ii) fragment position changes over time are captured either

by causal relations or constraint relations. Section 4.2 discussed how causal relations can change attribute values. The same approach is applied here so that a fragment position can change in reaction to an event.

By representing fragment position and size the same way as document variables, $\mathcal{SHM}$ also associates to them state-attributes, which shall indicate when their value changes. Moreover, $\mathcal{SHM}$ enables associating an attribute value to a time interval, so that an attribute ω has its value equal to v inside an interval i, similar to what is done in Belouaer and Maris [5]. Value v can be defined as a discrete value or as a polynomial of order $n > 1$, with the time t_i as the polynomial's variable. Time value t_i represents the amount of time since interval i begins.

Considering the example shown in Figures 2 and 3, the following equations represent changes in `vid`'s size when `ico` is selected (Equations (19–22)) and `fig`'s animation during its presentation (Equations (23–24)). We consider here a screen size of 640×480.

$$\langle 4,\ \Box ico_{(s,s)},\ \circ vid_{(x,c)} = 160,\ ico_{(s,e)} == true\rangle \tag{19}$$

$$\langle 5,\ \Box ico_{(s,s)},\ \circ vid_{(y,c)} = 120,\ ico_{(s,e)} == true\rangle \tag{20}$$

$$\langle 6,\ \Box ico_{(s,s)},\ \circ vid_{(x,s)} = 320,\ ico_{(s,e)} == true\rangle \tag{21}$$

$$\langle 7,\ \Box ico_{(s,s)},\ \circ vid_{(y,s)} = 240,\ ico_{(s,e)} == true\rangle \tag{22}$$

$$\langle fig_{(x,c)} = 160.cos(\alpha) + 320,\ \triangleright fig_{(p,s)} \leq t \leq \Box fig_{(p,s)}\rangle \tag{23}$$

$$\langle fig_{(y,c)} = -120.cos(\alpha) + 240,\ \triangleright fig_{(p,s)} \leq t \leq \Box fig_{(p,s)}\rangle, \tag{24}$$

$$\alpha = \frac{2\pi(t - \triangleright fig_{(p,s)})}{t_4}$$

where t represents the time since the document execution begins and t_4 is the duration of `fig`'s animation (see Figure 3(a)). In this example, a trigonometric polynomial was used for representing `fig`'s movement.

By using relations, $\mathcal{SHM}$ represents changes in a document's spatial layout in relation to time. Moreover, the use of a polynomial for defining the value of a positioning attribute is interesting since it brings with it the possibility of representing complex animations such as the ones provided by SMIL/SVG animation [38, 39] or Web Animations [41]. In such an approach, animation paths, keyframes, and so on are represented by polynomial interpolation.

5 DESCRIBING THE AUTHOR'S EXPECTATION

Multimedia documents can be created by an author with a range of available authoring tools. Depending on the language in use, tools may be able to provide the author with different views of the document, in addition to a preview of its spatio-temporal layout [29]. As presented in Section 4, $\mathcal{SHM}$ allows representing multimedia documents as a spatio-temporal layout ℓ.

As previously defined, validation over document d is carried against a set of tests $\mathcal{T}$. It is achieved by validating $\ell \models \mathcal{T}$, where ℓ is the $\mathcal{SHM}$ spatio-temporal layout obtained from document d. In this section, we discuss how an author may represent tests to be performed over an $\mathcal{SHM}$ spatio-temporal layout. As discussed previously, each test may represent an *expected* or an *unexpected* behavior that an author wants to verify in a document specification.

In previous works [14, 15], we formalized a set of predefined expected behaviors: *reachability* (will a media be presented?), *media termination* (given that a media is presented, does it end?), and *document termination* (does the document as a whole end?).

In order to enable an author to describe specific tests over a document, we define a set of atomic formulas in either the temporal or the spatial axis. The description of the expected spatio-temporal layout, therefore, is achieved by combining temporal and spatial atomic formulas. Moreover, such

Fig. 6. Allen's relations between time intervals.

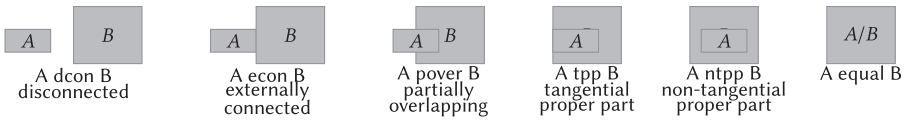


Fig. 7. RCC spatial relations between active regions.

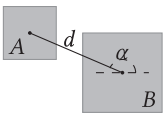


Fig. 8. Angle and distance between regions.

atomic formulas are specified in a high level of abstraction, which facilitates author description. It is important to highlight that the author is able to create the test set that he or she finds necessary for each document. Therefore, the test set's completeness is left for the author.

In the temporal axis, formulas represent Allen's relations [2] between time intervals. In Figure 6, we recall the set of Allen's relations, where rectangles represent time intervals.

In the spatial axis, formulas represent Randal, Cui, and Cohn (RCC) spatial relations between regions [35] as presented in Figure 7, where rectangles represent regions.

As discussed in dos Santos et al. [17], in order to enable the description of relative positions between regions and region distance, we extended RCC spatial relations (except *equal*) so that relations are parameterized by an angle between region centers. Moreover, relations *dcon*, *pover*, and *ntpp* are also parameterized by a distance between region centers. The angle and distance between two regions are calculated as presented in Figure 8. It presents two *disconnected* regions, A and B, with A being at an angle α and distance d with respect to B.

We represent such an example by the formula $A\ dcon(\alpha, d)\ B$. Angle α can be described in degrees or, if such precision is not necessary, by a(n) (intra)cardinal direction. Thus, the example of Figure 8 may also be represented by the formula $A\ dcon(NW, d)\ B$. If the direction information is given as a(n) (intra)cardinal point, the angle is given as follows:

$$\begin{aligned} &E = [337.5, 22.5),\ NE = [22.5, 67.5),\ N = [67.5, 112.5),\ NW = [112.5, 157.5),\\ &W = [157.5, 202.5),\ SW = [202.5, 247.5),\ S = [247.5, 292.5),\ SE = [292.5, 337.5). \end{aligned} \tag{25}$$

If a precise distance or an angle is not necessary, both d and α may be omitted. Also, it is worth noticing that the above formulas may be combined to create a more complex test. For example, the author may define a test such as

$$\texttt{fig}\ \textit{during}\ \texttt{vid}\ \rightarrow\ \texttt{fig}\ \textit{dcon}\ \texttt{vid}, \tag{26}$$

which states that if `vid` and `fig` are presented together, they should not overlap.

6 HYBRID VALIDATION APPROACH

The validation approach presented here is based on model $\mathcal{SHM}$, which is used for representing a document's spatio-temporal layout. Given its abstract representation of a document layout, it can be used for validating documents described with different declarative authoring languages.

As shown in Section 4, from model $\mathcal{SHM}$ it is possible not only to reason about document fragments in terms of their state along the document execution, but also about their projections in the temporal axis (as intervals). $\mathcal{SHM}$ allows the description of relations as both causal relations over event occurrences or as constraints over intervals.

For validation purposes, therefore, we may get from $\mathcal{SHM}$ two representations of a document: one in terms of its state along the document execution and another in terms of its intervals. We consider that the document state comprises the value of all $\mathcal{SHM}$ attributes associated with document fragments. To take advantage of both possible representations for a document, the hybrid validation approach we propose in this work is composed of two validation techniques, one for each representation.

The first technique, *docstate*, produces from the $\mathcal{SHM}$ model a representation of a document in terms of its states. The resulting representation is described into a rewriting theory [15–17], where document fragments are represented as state machines, and causal relations are modeled as equations that change their state along document presentation. Moreover, temporal constraints are modeled as linear temporal logic formulas, and validation is performed through model-checking. Section 6.1 describes this in more details.

The second technique, *docinterval*, produces from the $\mathcal{SHM}$ model a representation of a document in terms of intervals. The resulting representation is described as a set of SAT Modulo Theories (SMT) formulas, where document fragments are represented as projections in the temporal and spatial axes along document presentation. Both causal relations and constraints are represented as SMT assertions, and validation is performed through SMT solving [12]. Section 6.2 describes this in more details.

6.1 Document State–Based Validation

This section briefly describes the *docstate* validation technique. It captures an SHM spatio-temporal layout ℓ as a rewrite theory [33] $\mathcal{R}$.

Rewrite Theory $\mathcal{R} = (\Sigma, E, R)$ represents a layout ℓ in terms of its state along execution. Σ provides constructors for representing the document state, which is comprised by the set of all attributes; E represents the behavior of parts of the document in terms of relations, and R represents the (possibly) nondeterministic behavior of a document as a whole. Intuitively, R captures the nondeterminism induced by viewer interaction and time lapse. For a given document d, rewrite theory $\mathcal{R}$ is extended with information about d's attributes and rewrite rules modeling d's relations. This extension process is automated by transformation $\tau_{docstate}$. The extended theory is denoted by notation $\mathcal{R}_d$.

Every $\mathcal{SHM}$ attribute associated with a given fragment is represented in Σ as an element of sort *UnitAtt*. For a state-attribute, $\mathcal{R}$ declares operator `state`, which behaves as described in Section 4.1. The behavior for operator `state` is captured in E as a set of equations.

Σ also defines elements of sort *EventOccurrence* for signaling state changes. Every time a state-attribute changes its value, by the equations in E, an element of sort *EventOccurrence* is included in the term representing the document state as a whole. Relations, therefore, are represented as equations in E, whose left-hand side is composed of *EventOccurrence* and *UnitAtt* elements, while the right-hand side defines actions to be performed over attributes. The optional precondition of relations is captured by the equation as a test to be performed before the equation is applied.

Example relation 15's representation in $\mathcal{R}$ is presented in Listing 1.

```
1   eq [r15] : end('ico, sel) occur('ico, sel, n:Nat) = occur('ico, sel, n:Nat + 1) .
```

Listing 1 Relation 15 in $\mathcal{R}$

In the example, `n:Nat` is a variable that represents the value before the equation application. The same value is used on the right-hand side of the equation, so that the fragment value is incremented by 1. It is worth highlighting that (i) whenever a relation is applied, it consumes an *event occurrence*; and (ii) the application of a relation may trigger the application of another relation.

The behavior of the document as a whole in $\mathcal{R}$ is given by R with rules. Rule `step` performs a temporal progression in the document execution. Viewer interactions, on the other hand, may occur at any moment a given fragment is being presented. To simulate viewer interactions, for each fragment f that the viewer may interact with, auxiliary attributes are created to represent delays from the beginning of f presentation to the moment the viewer interacted with it. The number of auxiliary attributes to be created is defined by the validation user. Rules for viewer interaction, therefore, will trigger the selection of f whenever one auxiliary attribute delay ends.

Theory $\mathcal{R}$ is represented as a Maude [11] module `R`, which represents the general behavior for multimedia documents. The complete description of module `R` is presented in dos Santos [13]. For each particular document, we declare another module that extends theory $\mathcal{R}$ defining the document's initial configuration in terms of its attributes, relations, interaction rules, and an initial action. The document's initial configuration and initial action for starting the document execution are declared by equations that use operators `doc` and `ini`. Document relations are specified as equations, as seen in Listing 1.

The document's initial configuration is represented in Maude as a term. Maude allows us to either *rewrite*, *search,* or *modelCheck* a given term. The *rewrite* command chooses one of the possible traces of a term rewrite. However, opposite to equations, rules may be nondeterministic, thus producing concurrent rewrites. In Maude, we can search in the "space state" of a term rewrite using command *search*. The Maude model checker is executed through command *modelCheck*. This command receives a term and a temporal logic formula to be evaluated. It takes into consideration the same "state space" of the search command. Whenever a formula does not hold, the model checker produces a counterexample presenting a trace where the given formula does not hold.

Therefore, we can validate the behavior of a document in three possible ways: (i) using the *rewrite* command to simulate document d, presenting one possible presentation for it; (ii) using the *search* command to present all possible presentations of d or to present traces that lead the document state to the pattern defined in the search command; or (iii) using the *modelCheck* command to verify the existence of a given behavior in every possible execution of d.

The description of tests for expected or unexpected behaviors is done using $\mathcal{SHM}$'s temporal and spatial atomic formulas (see Section 5). Those formulas are formalized in *docstate* as LTL [34] (Linear Temporal Logic) formulas. It is worth noting that formulas in the temporal axis describe the evolution of (part of) the document state throughout several states. For example, formula *A meets B* is described by the following LTL formula:

```
1  eq A meets B = <> ( (state(A,pre,occurring) /\ state(B,pre,sleeping))
2                      /\ O (state(A,pre,sleeping) /\ state(B,pre,occurring)) ) .
```

where `<>` and `O` represent LTL operators F and X in Maude. The former states that a formula must be valid for some future state, and the latter states that a formula must be valid for the next state; notation `state(f,t,s)` means that fragment f is in state s, with t meaning its *presentation*, *selection,* or *attribution*. For example, *state*(*A*, *pre*, *occurring*) means that A presentation state is *occurring*.

On the other hand, formulas in the spatial axis consider position values for a given document state. They are combined with an LTL temporal operator for representing a spatio-temporal layout. As a usage example of a spatial formula, suppose we want to verify if, at all times, A and B will

be disconnected in space. Thus we write the following formula, $G(A\ dcon\ B)$, where we combine spatial relation *dcon* with temporal operator G, which states that a formula must be valid for all states. Formula *dcon* is described by the following LTL formula:

```
eq A dcon B = state(A,pre,occurring) /\ state(B,pre,occurring) /\
              (left(A) > left(B) + width(B) \/ left(A) + width(A) < left(B) \/
              top(A) > top(B) + height(B) \/ top(A) + height(A) < top(B)) .
```

where notation `left(f)` means the left position of fragment f and the same for the remaining position information.

In case the formula is parameterized by an angle, function *angle* is called together with the formulas presented previously to determine the angle between region centers. Function *angle* is parameterized by the expected angle in either degrees or one (intra)cardinal direction.

6.2 Interval-Based Validation

This section briefly describes the *docinterval* validation technique. It captures an $\mathcal{SHM}$ spatio-temporal layout ℓ as a satisfaction problem $\mathcal{S}$ using SMT [12]. SMT is a satisfaction problem where formulas combine logical connectives, such as conjunction, disjunction, and negation, with atomic formulas in the form of linear arithmetic inequalities. Formulas are composed by variables, representing either Boolean or arithmetic values. A solution for such a formula is an assignment, mapping variables to values that make the formula *true*.

Satisfaction problem $\mathcal{S}$ represents attributes as constants. Attribute $f_{(\alpha,d)}$ ($\alpha \in \{p, s, a\}$) is represented by constant $f_\alpha{:}t^{exp}$, attribute $f_{(\alpha,e)}$ by constant $f_\alpha{:}t^{plc}$, and state-attribute $f_{(\alpha,s)}$ is represented by its projection in the time using constants $f_\alpha{:}t^{beg}$ for its projection to begin, $f_\alpha{:}t^{mid}$ for its projection center, and $f_\alpha{:}t^{end}$ for its projection end. It is worth noting that attributes representing a fragment in space are translated to constants similar to those previously defined for the state projection.

In addition to constant $f_\alpha{:}t^{plc}$, $\mathcal{S}$ defines constant $f{:}s^{plc}$ to indicate whether or not a fragment is being presented on the screen. The relation between both constants $f{:}t^{plc}$ and $f{:}s^{plc}$ is given as follows:

$$\begin{aligned}&\left\{f{:}t^{plc} \ \wedge\ f{:}t^{beg} \leq t \leq f{:}t^{end} \ \wedge\ f{:}s^{plc}\right\} \ \vee \\ &\quad\left\{\left(\neg f{:}t^{plc} \ \vee\ t < f{:}t^{beg} \ \vee\ t > f{:}t^{end}\right) \ \wedge\ \neg f{:}s^{plc}\right\}\end{aligned} \tag{27}$$

where constant t is used to represent the time since the document execution begins. Equation (27) has the following meaning. Whenever a fragment is part of the temporal layout (i.e., $f{:}t^{plc} = true$), constant $f{:}s^{plc}$ shall be *true* while its state is equal to *occurring* and *false* elsewhere.

As discussed in Section 4.1, fragments may be paused during their presentation. As can be seen in Figure 5(b), the value of state-attribute ω is in the *paused* state inside an interval between $\text{▯}\omega$ and $\text{▶▶}\omega$. $\mathcal{S}$ represents when a state-attribute ω value is equal to *paused* by a second interval ω'. Both intervals are related as follows:

$$\omega{:}t^{beg} < \omega'{:}t^{beg} \ \wedge\ \omega'{:}t^{end} \leq \omega{:}t^{end} \tag{28}$$

$$\omega'{:}t^{plc} \rightarrow \omega{:}t^{plc} \tag{29}$$

$$(\omega'{:}t^{plc} \wedge \omega'{:}t^{exp} > 0) \ \vee\ (\neg\omega'{:}t^{plc} \wedge \omega'{:}t^{exp} = 0) \tag{30}$$

$$\omega{:}t^{exp} = v + \omega'{:}t^{exp} \tag{31}$$

which means that a given interval (ω') has to be inside the interval for ω (Equation (28)); if ω' exists, so does ω (Equation (29)); the expected size of ω' is greater than zero if it exists or equal

to zero in the opposite case (Equation (30)); and that the size of ω is its expected size plus ω' size (Equation (31)).

It is worth noting that the constants used for representing document fragments in either space or time introduce some redundancy. Some examples of redundant constants are $f{:}a^{mid}$, $f{:}a^{exp}$, and $f{:}s^{plc}$. However, this redundancy was included in order to ease the definition of expressions.

As seen in Section 4.1, $\mathcal{SHM}$ enables defining both causal relations and constraint relations. $\mathcal{SHM}$ relations are represented in $\mathcal{S}$ by assertions, which may include inequalities among the constants. Constraint relations have a straightforward representation in $\mathcal{S}$. Causal relations, however, are represented by assertions that compare event occurrence times for each event that triggers a same state change and chooses the first one to occur. For example, consider the causal relation γ_3 in Equation (10). It ends `ico` whenever the viewer interacts with it (i.e., attribute $ico_{(s,e)}$ is *true*). Relation γ_3 is represented in $\mathcal{S}$ by the following constraint:

$$\{ico_s{:}t^{plc} \wedge (ico_s{:}t^{end} < ico_p{:}t^{beg} + ico_p{:}t^{exp}) \wedge ico_p{:}t^{end} = ico_s{:}t^{end}\} \vee$$
$$\{(ico_p{:}t^{beg} + ico_p{:}t^{exp} \leq ico_s{:}t^{end} \vee \neg ico_s{:}t^{plc}) \wedge ico_p{:}t^{end} = ico_p{:}t^{beg} + (ico_p{:}t^{exp})\} \quad (32)$$

Equation (32) means that the projection for attribute $ico_{(p,s)}$ will end with the first one to occur: (i) viewer interaction (i.e., the end of $ico_{(s,s)}$) or (ii) its end by its expected duration when it has an intrinsic duration or the infinite value (constant I) otherwise.

$\mathcal{S}$ represents variable value changes in the same way as presented in Section 4.4. Thus, the value of a given constant will be associated with a given interval, such that its value will be equal to v inside that interval and some other value (possibly the same) outside the interval. $\mathcal{S}$ supports defining value v either as a number or a polynomial function over constant T.

In order to provide validation for a given document d, its representation as a satisfaction problem ($\mathcal{S}_d$) is fed into an SMT solver. The solver builds the formula representing the whole satisfaction problem from the defined *assertions* as a conjunction of them. The solver then tries to solve the problem iterating through *assertions* and assigning values to the constants in *assertions*. The goal is to keep each assertion evaluated to *true*. Whenever an *assertion* does not hold, the solver backtracks and starts again, assigning new values to constants. At the end of this process, when the whole formula holds, the solver presents back the achieved valuation for constants declared in $\mathcal{S}_d$. Validation using SMT can be done in the following ways.

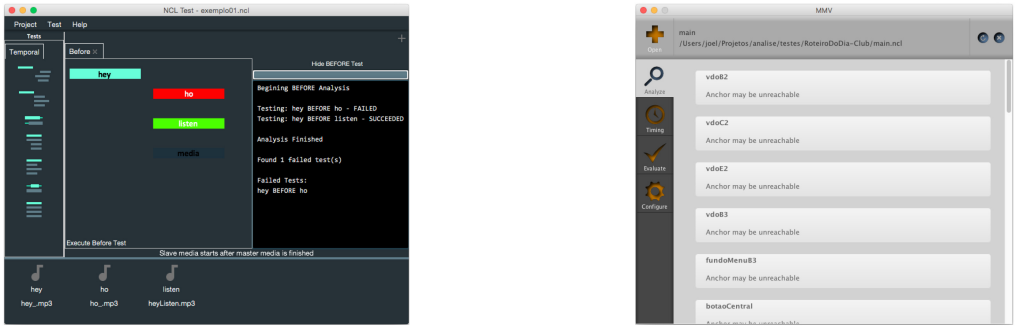
The first way to provide validation is by checking the possibility of evaluating $\mathcal{S}_d$. Given that no valuation is provided by the SMT solver, it is possible to infer that the set of relations in d is inconsistent. It is worth mentioning that this approach is commonly used during document authoring, as a new relation is included in the document layout. Whenever validation returns an error, it is possible to infer that the new relation made the document layout inconsistent.

The second way to provide validation is by computing conjunction $\mathcal{S}_d \wedge \mathcal{T}$, where $\mathcal{T}$ is a test representing an *expected* or *unexpected* behavior. Given that a valuation is provided by the solver, it is possible to infer that a given test $\mathcal{T}$ holds, and the document presents the tested behavior.

6.3 Implementation

In this section, we present our hybrid validation approach implementation. Both *docstate* and *docinterval* validation techniques have been implemented into validation tools.

`aNaa` is an API built for use by other tools that focus on NCL [26] document authoring. It extends API *aNa* [18] with methods to create the representation of a given document in $\mathcal{R}$ and call Maude to perform validation using the Maude model-checker tool.



(a) NCL-Tester (b) aNaa4Web

Fig. 9. aNaa-based tools interface.

In a nutshell, aNaa transforms NCL documents into $\mathcal{R}$ as follows. (i) For each NCL *media* and *context* element (representing media items and compositions, respectively), a set of attributes is produced representing it in time and space. The same is done for each *area* and *property* element (representing a media item subpart and a media item variable, respectively). (ii) For each *media*, *area*, and *context* element, its implicit temporal relations defined by NCL produce equations that change their state accordingly. For example, an NCL *context* is in the *occurring* state if at least one of its inner elements is in the *occurring* state. For each *context/media* pair, an equation is produced to mimic such behavior. The same is true for *media/area* and *context/context* pairs. (iii) NCL causal relations (*links*) are manipulated such that a relation that points to element *A*, which in turn points to element *B*, will directly point to element *B*. (iv) *Links* are simplified [31] until they have only one event occurrence as condition, and *links* with the same condition are joined into one. For every resulting `link`, a Maude equation (see Listing 1) is produced.

Although aNaa is implemented for validating NCL documents, we have also implemented a transformation for SMIL to *docstate*. It is worth mentioning, however, that such implementation is not yet integrated to *aNaa*. Such integration was left for future work. A detailed description of both transformations is presented in dos Santos [13].

Two tools were built on top of aNaa, named `NCL-Tester` and aNaa4Web. `NCL-Tester` [4] (Figure 9(a)) provides to the author a graphical way for defining tests based on the relations presented in Section 5 and perceiving the validation result. Once several tests are created, the user has the option to validate a single test or the test suite as a whole. aNaa4Web is a user interface built on top of aNaa (Figure 9(b)). It provides validation of the document syntax and its behavior. Validation in aNaa4Web is performed against a set of predefined tests briefly presented in Section 5.

`SMT Validator` is a tool implemented for performing the *docinterval* validation technique presented in Section 6.2. The validator tool is implemented in Lua on top of the Yices2 [19] solver. It provides functions encapsulating the operations used for representing document elements and relations. It also provides functions for controlling the Yices2 solver, such as creating a context, backtracking, checking the context, and evaluating constants.

It is worth highlighting that the implementation of the `SMT Validator` in Lua brings the possibility of integrating it with an NCL player (as a module of the NCL document itself). An application of such a scenario is presented in Amorim et al. [3], where it is used for calculating, during execution, media item position from a set of spatial constraints. It is worth mentioning that media items' positions may change along document execution as they start or stop their presentations.

The idea of the hybrid approach presented in this article is to provide the translation from a given multimedia language L_i to $\mathcal{SHM}$ and from the latter to either *docstate* or *docinterval* in order

Table 2. Tools Implementation Characteristics

	aNaa	aNaa4Web	NCL-Tester	SMT Validator*
Target language	NCL, SMIL	NCL	NCL	independent
Type of tests	predefined, user-defined	predefined	user-defined	user-defined
Validation type	*docstate*			*docinterval*
Translation	NCL → *docstate*			SHM → *docinterval*

to provide document validation. Currently, our validation tools implement part of the proposed approach. Table 2 summarizes the target authoring language, type of tests, validation technique, and translation step implemented by each validation tool.

aNaa provides a mapping from NCL (or SMIL) to *docstate*. SMT Validator, on the other hand, provides a mapping from $\mathcal{SHM}$ (or at least from the Lua table representing it) to *docinterval*. Although we have already explored mappings from NCL to $\mathcal{SHM}$, it is not yet automated.

Table 2 presents four tools. aNaa, aNaa4web, and NCL-Tester have already been discussed in previous works [4, 14–17]. SMT Validator is proposed in this work. It explores the *docinterval* validation, a feature not explored in previous works. A limitation regarding the current implementation, therefore, is the lack of integration of both approaches. A future work in this direction is to provide a representation of $\mathcal{SHM}$ in either Java and/or Lua so that we can translate a given document to it and then to either *docstate* or *docinterval* representations.

6.4 Experimentation

In this section, we present practical results about the use of the *docstate* validation technique. Given that the translation from NCL documents to *docinterval* is not automated yet, usability tests with SMT Validator are left for future work.

As an indication of a reasonable performance of *docstate*, aNaa performance was evaluated for the validation of real documents found in the NCL Club[1] repository. Document **"First João"** synchronizes a video with photos. It allows the viewer to interact with one of the photos it presents. If an interaction occurs, the video is resized, and a second video starts playing. Document **"Live More"** presents a TV show where the viewer may interact to choose one among four meal options presented by the show presenter. Once the choice is made, additional information about each option is provided. Document **"Day's Route"** is a nonlinear show where the user can define the narrative line by choosing the next sight in a city tour.

The above documents were chosen given their different sizes and interaction possibilities. They were validated using aNaa and the set of predefined tests presented in Section 5. The average time spent for validating a predefined test in each of the three documents was 2.25*ms*, 1.83*ms*, and 2.58*ms*, respectively. Tests were conducted in an Intel i5 computer with 16GB memory.

To indicate the reasonable performance of our tool, we compared our results with the tool proposed in Picinin et al. [28]. The latter was used for the analysis of **"First João"** and **"Live More"** documents. The same behavior identified by our tool was identified using that tool. The time spent, however, was around 15s and 40s, respectively.[2]

NCL-Tester was evaluated by users in two different usability tests. The first test was conducted with 11 test subjects and was discussed in Barreto et al. [4]. A new round of tests involved 12 test subjects. Test subjects were enrolled in Computer Science courses, either graduate or undergraduate. In both tests, subjects had different NCL proficiency levels. In order to measure their

[1] http://club.ncl.org.br.
[2] Information kindly provided by the authors of Picinin et al. [28].

Table 3. Second Test Results

	Code Inspection			NCL-Tester		
	Naive	**Expert**	**All**	**Naive**	**Expert**	**All**
Subjects	3	3	6	4	2	6
Duration (minutes)	$\mu = 11$ $\sigma = 5.29$	$\mu = 14.66$ $\sigma = 10.26$	$\mu = 12.83$ $\sigma = 7.57$	$\mu = 42$ $\sigma = 24.55$	$\mu = 25$ $\sigma = 7.07$	$\mu = 36.33$ $\sigma = 21.18$
Errors found	$\mu = 1.33$ $\sigma = 1.15$	$\mu = 2.33$ $\sigma = 1.53$	$\mu = 1.83$ $\sigma = 1.33$	$\mu = 2$ $\sigma = 0.82$	$\mu = 3$ $\sigma = 1.41$	$\mu = 2.33$ $\sigma = 1.03$

proficiency in NCL, test subjects had to answer a question evaluating proficiency from 1 to 5 in the first test, with those answering 2 or less being considered naive. Proficiency was rated as 1 to 6 in the second test, with those answering 3 or less being considered naive.

In the first test, 6 subjects were considered expert and 5 were considered naive. Subjects received four different documents and a detailed description of their expected behavior. They had to create tests, one for each expected behavior, and validate if the document behavior would fit the description. After performing the tasks, they evaluated `NCL-Tester` according to the System Usability Scale (SUS) [10]. The tool SUS score was 68.4, slightly above the limit considered acceptable (68). However, when considering just the expert users, its score was 75.4. Such a difference may be explained by the fact that expert users are familiar with the test graphical notation used by the tool, which is common in other multimedia document authoring tools.

In the second test, 5 subjects were considered expert and 7 were considered naive. Subjects received a detailed text description about the expected spatio-temporal layout of an application and an NCL document that was supposed to implement it. This document, however, has 5 authoring errors. Test subjects were divided into two groups at random. The first group had to check the application by executing it and/or XML code inspection. The second group had to validate the application using `NCL-Tester`. After performing the tasks, they had to answer a questionnaire indicating test duration, number of errors found, and specific questions about `NCL-Tester`. Test results are presented in Table 3.

On average, subjects using `NCL-Tester` took more time to validate the application than did those doing it by coding inspection. Since there was no presentation about `NCL-Tester` and how to perform the tests, subjects reported they needed time to understand the tool interface before actually performing the validation. This can be seen in the difference between the time taken by naive subjects and the expert ones using the tool. One should note that the latter are familiar with the notation used by the tool, thus explaining the shorter times. Considering the number of errors found, subjects using `NCL-Tester` found on average more errors than did those who did code inspection. Naive subjects using `NCL-Tester` were able to find approximately the same number of errors as expert ones using code inspection.

`NCL-Tester` specific questions were used to evaluate how easy one creates, changes, and understands tests. Each question was evaluated from 1 to 6. Subjects found it easy to create tests ($\mu = 5.17$, $\sigma = 0.98$), change an existing test ($\mu = 5.33$, $\sigma = 0.52$), and understand the results of a test ($\mu = 4.83$, $\sigma = 1.60$). The latter was better evaluated by expert subjects ($\mu = 6$, $\sigma = 0$) than naive ones ($\mu = 4.25$, $\sigma = 1.71$). Moreover, subjects had little difficulty in understanding the purpose of a test ($\mu = 3.5$, $\sigma = 1.05$).

Although users found the tool easy to use, test results indicate that changes in the tool interface would be welcome, especially in the representation of atomic tests. However, considering that the focus of this article is the validation approach itself, we considered that it presented good results since it helped users find more errors in NCL documents.

7 VALIDATION TECHNIQUE CHOICE

The hybrid validation approach we present in this article is based on the application of two validation techniques, *docstate* and *docinterval*. Each technique is capable of representing a given aspect of $\mathcal{SHM}$. *docstate* focuses on event occurrences and the document state throughout its presentation, being more suited for validating documents where events and different occurrences of fragments are an important feature. *docinterval* focuses on numeric dependencies among intervals (or regions) and is more suited for validating documents where unknown time values and continuous changes are an important feature.

Technique *docstate* works on the complete representation of a document d, generating all states of its presentation. In addition, due to its event-based nature, it allows for investigating properties related to different occurrences of a given fragment along document execution.

As a *docstate* usage example, let us consider a modification in the example presented in Figures 2 and 3, such that `vid` may be *paused* and *resumed* as many times as the viewer wants by interacting with it. In that case, suppose we want to investigate if, whenever `vid`'s presentation is *paused*, it can be *resumed*. Encoding such a test in *docstate* results in the following LTL formula:

```
T1 : []( state('vid,pre,paused) -> <> state('vid,pre,occurring) )
```

where [] represents the LTL operator G in Maude.

Technique *docstate* performs better than *docinterval* in such a scenario. Since *docinterval* represents state-attributes as intervals, one interval must be created for each time it goes to the *occurring* state; that is, for each occurrence $f_i, i \in [1, n]$ of a given fragment f along the presentation. If n is not known *a priori*, using *docinterval* would be cumbersome. Moreover, representing the above property in *docinterval* is not an easy task.

Differing from *docstate*, technique *docinterval* does not expect the complete representation of d, thus it is able to validate it even when d represents a partial document. Due to its numerical nature, *docstate* allows investigating properties related to (i) the duration of fragments, (ii) the relative position of event occurrences in time, and (iii) the relative position of regions in space.

As a *docinterval* usage example, we may investigate in the example if it is possible for `vid` and `fig` to end their presentation at the same time. Encoding such a test in *docinterval* results in the following SMT formula:

$$\mathtt{T2}{:}vid_p{:}t^{plc} \wedge fig_p{:}t^{plc} \wedge vid_p{:}t^{end} = fig_p{:}t^{end} \tag{33}$$

By solving $\mathcal{S}_d \cap \mathtt{T2}$ in the SMT solver, we are able to verify that such a situation is not possible since, while `fig` is presented, `vid` is paused, and it is not possible to start `fig` immediately before it ends.

Technique *docinterval* performs better than *docstate* in such a scenario. Given that *docstate* represents the document presentation by its different states along execution,[3] simulating viewer interaction at different time instants would require generating every possible combination of viewer interaction in order to search for a possible path where both `vid` and `fig` end at the same time. Thus, it leads to an explosion in the number of states created by *docstate*.

Due to the different characteristics of *docstate* and *docinterval*, each technique complements the other in terms of the expressiveness of $\mathcal{SHM}$ and the kind of tests to be investigated. Using a combination of both techniques for document validation would improve the range of tests to be performed over a given document. The hybrid validation approach we present in this article implements both techniques, as presented in Section 6.3.

[3]Different traces are provided by the interleaving of rules *step* and *interact* as discussed in Section 6.1.

8 CONCLUSION

Although the use of a declarative authoring language may ease the authoring effort, it is still possible that the resulting spatio-temporal layout does not fit the author's expectations due to the incorrect use of constructions available in the authoring language being used. This article discussed the use of validation to avoid such mismatch.

Our work proposed the spatio-temporal validation of multimedia documents based on a hybrid approach. It relies on a formal model called $\mathcal{SHM}$. From a document representation in $\mathcal{SHM}$, two validation techniques named *docstate* and *docinterval* may be applied. *docstate* captures the behavior of a document in terms of its state throughout its execution. It is based on a rewrite theory that induces a transition system where states represent the document state and transitions model user interactions or time lapse. Validation in *docstate* is performed by model-checking. *docinterval* captures the behavior of a document in terms of intervals and event occurrences. Document fragment intervals and regions, and relations among fragments, are described through SMT formulas. Validation in *docinterval* is performed by solving the resulting formulas in an SMT solver. Atomic formulas are available to the author for creating the tests necessary for a given document. Such an approach favors the completeness of the test set for a given document to be validated.

Due to the different characteristics of *docstate* and *docinterval*, each technique complements the other in terms of the expressiveness of $\mathcal{SHM}$ and the kind of properties to be investigated. Currently, the choice for the validation technique to be used is left to the author. A future work is to investigate an approach to automatically choose one of them based on the characteristics of the document and the tests to be validated.

The proposed approach is intended to be used with different authoring languages. In dos Santos et al. [16] we presented a formal proof that $\mathcal{SHM}$ captures the semantics of NCL. Proofs for other languages such as HTML5 and SMIL are future works.

The current implementation of *docinterval* uses Lua and the SMT solver Yices2 [19]. An ongoing work seeks to integrate it with NCL in order to provide dynamic layout adaptation for NCL presentations. Given that Lua is used as an auxiliary scripting language by NCL, we are able to execute the validation as part of an NCL document at runtime and modify the document's spatial layout according to the valuation provided by the solver. A future work is to integrate *docstate* as well to further investigate its use in such a scenario.

docstate and *docinterval* implementations were tested with real-world documents. Furthermore, usability tests were also conducted with *docstate*. Tests results are presented in dos Santos [13]. As a future work, we plan to run more tests to better evaluate our proposal.

Given a property to be validated over a document, both tools shown in Section 6.3 present back to the author if the property is valid or not. A future work is to present back to the author more detailed messages. One example is to use the information in the trace presented as an answer by the Maude model checker to build an animation or a sequence of NCL link activations up to the point of failure.

REFERENCES

[1] ABNT. 2011. Digital terrestrial television - Data coding and transmission specification for digital broadcasting - Part 2: Ginga-NCL for fixed and mobile receivers - XML application language for application coding. (2011). ABNT NBR 15606-2:2011 standard.

[2] J. F. Allen. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26, 11 (1983), 832–843.

[3] Glauco F. Amorim, Joel A. F. dos Santos, and Débora C. Muchaluat-Saade. 2016. XTemplate 4.0: Providing adaptive layouts and nested templates for hypermedia documents. In *Proceedings of the 22nd International Conference on Multimedia Modeling*. Springer, Miami, FL, USA, 642–653.

[4] F. Barreto, D. Tamaki, Joel A. F. dos Santos, and D. C. Muchaluat-Saade. 2016. NCL-tester: Graphic application for NCL documents temporal test creation. In *Proceedings of the 22th Brazilian Symposium on Multimedia and the Web (WebMedia'16)*. ACM, Teresina, Brazil, 91–94 [in Portuguese].

[5] L. Belouaer and F. Maris. 2012. SMT spatio-temporal planning. In *Proceedings of the ICAPS 2012 Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS 2012)*. 6–15.

[6] Elisa Bertino, Elena Ferrari, Andrea Perego, and Diego Santi. 2005. A constraint-based approach for the authoring of multi-topic multimedia presentations. In *Proceedings of the IEEE International Conference on Multimedia and Expo*. IEEE Computer Society, Amsterdam, Netherlands, 578–581.

[7] G. Blakowski and R. Steinmetz. 1996. A media synchronization survey: Reference model, specification and case studies. *Journal on Selected Areas in Communications* 14, 1 (January 1996), 5–35.

[8] Samia Bouyakoub and Abdelkader Belkhir. 2008. H-SMIL-net: A hierarchical petri net model for SMIL documents. In *Proceedings of the Tenth International Conference on Computer Modeling and Simulation (UKSIM'08)*. IEEE Computer Society, Washington, DC, 106–111.

[9] Samia Bouyakoub and Abdelkader Belkhir. 2011. SMIL builder: An incremental authoring tool for SMIL documents. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)* 7, 1 (Feb. 2011), 2:1–2:30.

[10] John Brooke. 1996. SUS-A quick and dirty usability scale. *Usability Evaluation in Industry* 189, 194 (1996), 4–7.

[11] M. Clavel, S. Eker, F. Durán, P. Lincoln, N. Martí-Oliet, and J. Meseguer. 2007. *All About Maude - A High-performance Logical Framework: How to Specify, Program, and Verify Systems in Rewriting Logic*. Vol. 4350. Springer-Verlag.

[12] Leonardo De Moura and Nikolaj Bjørner. 2011. Satisfiability modulo theories: Introduction and applications. *Communications of the ACM* 54, 9 (2011), 69–77. DOI: http://dx.doi.org/10.1145/1995376.1995394

[13] Joel A. F. dos Santos. 2016. *Multimedia Document Validation Along Its Life Cycle*. Ph.D. Dissertation. Universidade Federal Fluminense.

[14] Joel A. F. dos Santos, Christiano Braga, and Débora C Muchaluat-Saade. 2012. A model-driven approach for the analysis of multimedia document. In *SLE (Doctoral Symposium)*. Dresden, Germany, 37–44.

[15] Joel A. F. dos Santos, Christiano Braga, and Débora C Muchaluat-Saade. 2013. An executable semantics for a multimedia authoring language. In *Formal Methods: Foundations and Applications*, Juliano Iyoda and Leonardo de Moura (Eds.), Vol. 8195. Springer, Brasília, Brazil, 67–82.

[16] Joel A. F. dos Santos, Christiano Braga, and Débora C. Muchaluat-Saade. 2015a. A rewriting logic semantics for NCL. *Science of Computer Programming* 107–108 (2015), 64–92.

[17] Joel A. F. dos Santos, Christiano Braga, Débora C. Muchaluat-Saade, Cécile Roisin, and Nabil Layaïda. 2015b. Spatio-temporal validation of multimedia documents. In *Proceedings of the 2015 ACM Symposium on Document Engineering*. ACM, Lausanne, Switzerland, 133–142.

[18] Joel A. F. dos Santos, Julia V. Silva, Renan R. Vasconcelos, Wagner Schau, Cláudia Werner, and Débora C. Muchaluat-Saade. 2012. aNa: API for NCL authoring. In *Proceedings of the 18th Brazilian Symposium on Multimedia and the Web - Workshop of Tools and Applications*. ACM, São Paulo, Brazil.

[19] Bruno Dutertre. 2014. Yices 2.2. In *Computer-Aided Verification (CAV'2014) (Lecture Notes in Computer Science)*, Armin Biere and Roderick Bloem (Eds.), Vol. 8559. Springer, 737–744.

[20] S. Elias, K. S. Easwarakumar, and R. Chbeir. 2006. Dynamic consistency checking for temporal and spatial relations in multimedia presentations. In *Proceedings of the 2006 ACM Symposium on Applied Computing*. ACM, Dijon, France, 1380–1384.

[21] M. F. Felix. 2004. *Formal Analysis of Software Models Oriented by Architectural Abstractions*. Ph.D. Dissertation. Pontifícia Universidade Católica do Rio de Janeiro. in Portuguese.

[22] Ombretta Gaggi and Annalisa Bossi. 2011. Analysis and verification of SMIL documents. *Multimedia Systems* 17, 6 (April 2011), 487–506.

[23] Thomas R. G. Green and Marian Petre. 1996. Usability analysis of visual programming environments: A 'cognitive dimensions' framework. *Journal of Visual Languages & Computing* 7, 2 (1996), 131–174.

[24] Hazel Lynda Hardman. 1998. *Modeling and Authoring Hypermedia Documents*. Ph.D. Dissertation. Universität Amsterdam. Retrieved from https://homepages.cwi.nl/lynda/thesis/.

[25] Roberto Ierusalimschy. 2006. *Programming in Lua* (2nd ed.). Roberto Ierusalimschy.

[26] ITU. 2009. Nested Context Language (NCL) and Ginga-NCL for IPTV services. http://www.itu.int/rec/T-REC-H.761-200904-S. (2009). ITU-T Recommendation H.761.

[27] M. Jourdan, N. Layaida, C. Roisin, L. Sabry-Ismail, and L. Tardif. 1998. Madeus, an authoring environment for interactive multimedia documents. In *Proceedings of the 6th ACM International Conference on Multimedia*. ACM, Bristol, England, 267–272.

[28] Delcino Picinin Júnior, Jean-Marie Farines, and Cristian Koliver. 2012. An approach to verify live NCL applications. In *Proceedings of the 18th Brazilian Symposium on Multimedia and the Web*. ACM, Salvador, Brazil, 223–232.

[29] Ioannis Kostalas, T. Sellis, and Michalis Vazirgiannis. 1999. Spatiotemporal specification & verification of multimedia scenarios. In *Database Semantics*, Zahir Tari and Scott Stevens (Eds.), Vol. 11. Springer, 169–188.

[30] Sébastien Laborie, Jérôme Euzenat, and Nabil Layaïda. 2011. Semantic adaptation of multimedia documents. *Multimedia Tools and Applications* 55, 3 (2011), 379–398. DOI: http://dx.doi.org/10.1007/s11042-010-0552-9

[31] Guilherme Augusto Ferreira Lima and Luiz Fernando Gomes Soares. 2013. Two normal forms for link-connector pairs in NCL 3.0. In *Proceedings of the 19th Brazilian Symposium on Multimedia and the Web*. ACM, Salvador, Brazil, 201–204.

[32] Huadong Ma and K. G. Shin. 2004. Checking consistency in multimedia synchronization constraints. *IEEE Transactions on Multimedia* 6, 4 (Aug 2004), 565–574. DOI: http://dx.doi.org/10.1109/TMM.2004.830807

[33] J. Meseguer. 2012. Twenty years of rewriting logic. *The Journal of Logic and Algebraic Programming* 81, 7 (2012), 721–781.

[34] A. Pnueli. 1977. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, Washington, DC, USA, 46–57.

[35] David A. Randell, Zhan Cui, and Anthony G. Cohn. 1992. A spatial logic based on regions and connection. *Principles of Knowledge Representation and Reasoning* (1992), 165–176.

[36] L. F. G. Soares and R. F. Rodrigues. 2005. *Nested Context Model 3.0 Part 1 - NCM Core*. Technical Report. Informatics Department, PUC-Rio, Rio de Janeiro.

[37] Luiz Fernando G. Soares, Rogério Ferreira Rodrigues, Renato Cerqueira, and Simone Diniz Junqueira Barbosa. 2010. Variable and state handling in NCL. *Multimedia Tools and Applications* 50, 3 (2010), 465–489.

[38] W3C. 2008. Synchronized Multimedia Integration Language - SMIL 3.0 Specification. World-Wide Web Consortium Working Recommendation. Retrieved from http://www.w3c.org/TR/SMIL3.

[39] W3C. 2011. Scalable Vector Graphics (SVG) 1.1. World-Wide Web Consortium Working Recommendation Retrieved from http://www.w3.org/TR/SVG11.

[40] W3C. 2014. HTML5: A vocabulary and associated APIs for HTML and XHTML. World-Wide Web Consortium Candidate Recommendation. https://www.w3.org/TR/2014/REC-html5-20141028.

[41] W3C. 2014. Web Animations 1.0. World-Wide Web Consortium Working Draft. Retrieved from http://www.w3.org/TR/web-animations/.

Received May 2017; revised May 2018; accepted August 2018